Final Report

YuBin Ng Robert Zhou YunLan Tang Siqi Wei

June 11, 2014

1 Introduction

Our aim is to implement an ios game that involves database and web server back-end and a graphical user interface as a front-end, which is optimised for iphone. The game took inspiration from el clasico board game, Monopoly but takes place in the London underground. It is played between friends by visiting tube stations in real life. Who ever visited the station the most is the owner of the station and others who visited the station will have to pay the owner a certain amount of in-game money.

To play the game, the player simply create a game room, invite friends into the game and he will be able to see all the stations and players' information. The player will be able to enter at a station according to it's GPS location, and exist at a station. The server will calculate its route and process the transactions accordingly.

1.1 Feature List - Requirement

For our product to be have minimal requires functionality, it must:

1. Players must join the game by downloading the app and authorizing it with facebook.
2. Be able to store information in database, i.e. login and register
3. Be able to store the information about each game rooms created by users, i.e. players information, stations informations
4. Be able to update the information on server according to user input, i.e. when a user make a move, the server must be able to process the move, and all other client must have the latest game information.
5. Use one section of the Piccadilly line as the game map
6. Each visit to the station pay the owner 1 pound
7. Be able to exchange owner correctly, i.e. when the player has the most visit count of a station, it must become the owner

1.2 Feature List - Extra Feature

After the basic functionalities have been correctly implemented, we aim to achieve the following extra features to make the game more interesting

Completed by the report deadline:

1. Be able to see the profile picture of other players
2. Be able to logout from the game
3. Be able to let user exit a game room
4. Be able to invite friends who did not register with the game to play
5. Be able to have a variable payment at each station, which can be achieved by various mechanism
6. Be able to have quests, which the player will be rewarded if completed
7. Be able to have random events during each round, and the effect can be either negative or positive, i.e. increase/decrease of amount of money
8. Have a map view for user so that it can see the tube line
9. Be able to see information about other player in the game room
10. Have different game mode which the player can choose when creating a room

To be Completed in the future:

1. More choice of tube lines and stations
2. More choice of cities to play
3. To play with friends without a facebook account
4. Mini games to play during the tube journey to get rewards
5. Apple game center feature
6. Achievements

2 Project Management

2.1 Group Structure

We have divided up the work according to each group member's strength. Robert and YunLan are in charge of user interface (front-end) and graphic design. While Siqi Wei and YuBin Ng are in charge of server-side programming and database (back-end). We were all involved in game logic planning, design and implementation of features. We have used 'git' to update our files in the repository, and shared documents in 'google docs' for communication between members. We have broken down the whole program into small individual components and features and implement them step by step. Regular meetings were held in the lab to keep up with each other and discuss any issues or problems.

At the later stage of the project, Robert and Ken started to work together to integrate the front-end interface and back-end database, they worked together by defining a set of API calls that the front end can use to retrieve and send data to the server, and how each API should be implemented. At the same time, Yubin worked on an API to deal with GPS location and Yunlan worked on redesigning the UI of the game to make it more user friendly.

2.2 Implementation Language and Technology

Since we have evaluated the strength of each individual member, this has greatly influenced our implementation languages. We have chosen to work on languages we are comfortable with so that we can work smoothly and quickly on developing the game instead of spending too much time learning the language.

2.2.1 Objective-C

On the client side, we are using objective-C, this is the native language supported by the iOS platform with extensive online documentation and APIs, which makes the development process easier. The app will be more stable too.

2.2.2 PHP and PostgreSql

On the server side, server interface was implemented with PHP as it is free, open source and can be easily installed on all servers. It is widely used and has extensive documentations and supports online. PHP is a powerful web programming language that is being employed by most big websites on the internet such as Facebook and Twitter. PHP is easy to learn and it provides many built-in functions to achieve many things that a programmer would need. Furthermore, it is a language that is supported by all browsers and PHP interpreter can be installed easily on the server. Apart from PHP, we will incorporate the use of PostgreSQL to act as the back end database.

2.2.3 Heroku

Our group has chosen to use our own web server (Heroku) to host the game. This is because the college server does not have some server components installed. Heroku has a lot of support online and provides a lot of scalability in later stage of the project.

2.3 Design Processes

We seperate responsibilities across every member of the group. Everyone will be assigned a specific task. They will then come out with a draft for group discussion. In our regular meetings, we will give feedbacks and suggestions. We will then implement it individually after getting the group consent to avoid implementation conflicts with other people's work. For example, we make sure we agree on the return type of a particular function. After implementation, we will test each part individually before integrating it with the master branch.

2.4 Back-up systems

We mainly used 'git' version control system along with Heroku to back-up our project. Since it allowed us to keep track of previous working versions of the project, and prevented overwriting modified files when more than one person were updating them. We were able to back up data automatically as well as being sure that the changes each member has made will be seen by other members. Git also allow each user of the group repository to have a separate repository of his own, so that each member can have a copy of the project available on their directory (rather than relying solely on the group directory). Heroku is a reliable service so that our back-up will be safe on their server.

2.5 Testing

We have tested the game incrementally throughout the implementation phase. We started with setting up the database and user data management (e.g. sign-up, log-in), and tested whether all the data are dealt with correctly. Once the system and the basic frame of GUI are tested, we started to add one feature at a time. We made sure that each component we added worked as intended before proceeding with the next feature, so that we could always go back to the previous version if the implementation of the current feature was incomplete. This way of development also allowed us to concentrate on one feature at a time, and ensure loose coupling between the files (less dependency) while maintaining high cohesion within each file (achieving a common goal). This is a recommended design practices we have learnt from Software Engineering course. Apart from testing the implementation, we also tested the game flow and balance between users by putting ourselves in the players' position during the implementation phase. By actually playing the game and trying out implemented features ourselves, we were able to adjust the figures accordingly.

3 Program Description and Implementation

The name of our game is Metropoly. We have made use of all tube stations in London as our theme to make our game more engaging.

3.1 Design Patterns

We have employed various design pattern to aid out development throughout the project.

- MVC pattern, this was used throughout the development of the front-end iOS application, all data was stored using our model, i.e. GameRoom and GameStation, and the controller will receive interactions and send them to the model and make sure the view will display the latest correct data.
- Adaptor/Facade, this was used in the LocationUtil class, we need the functionality to know what are the closest stations to the device. The iOS SDK provided the CLLocationManeger class which has too many functionalities that we don't actually need, so I have created a LocationUtil, which hide the complexity of CLLocationManager and at the same time provide extra functionality which can give us an array of nearby Stations

- Facade-NetworkUtil For NetworkUtil class, the inner functionality of it is quite complicated as to initialise and to send synchronised request, however, with a facade pattern we hide those complex interactions, and therefore make the interface more clean.

- Factory-GameRoom When getting all the game rooms belong to a player, and when a player is creating a new room, we used a factory patter so that the user do not have to worry about how the GameRoom object was constructed, especially when retrieving all the rooms for a player, it will involves a huge complexity.

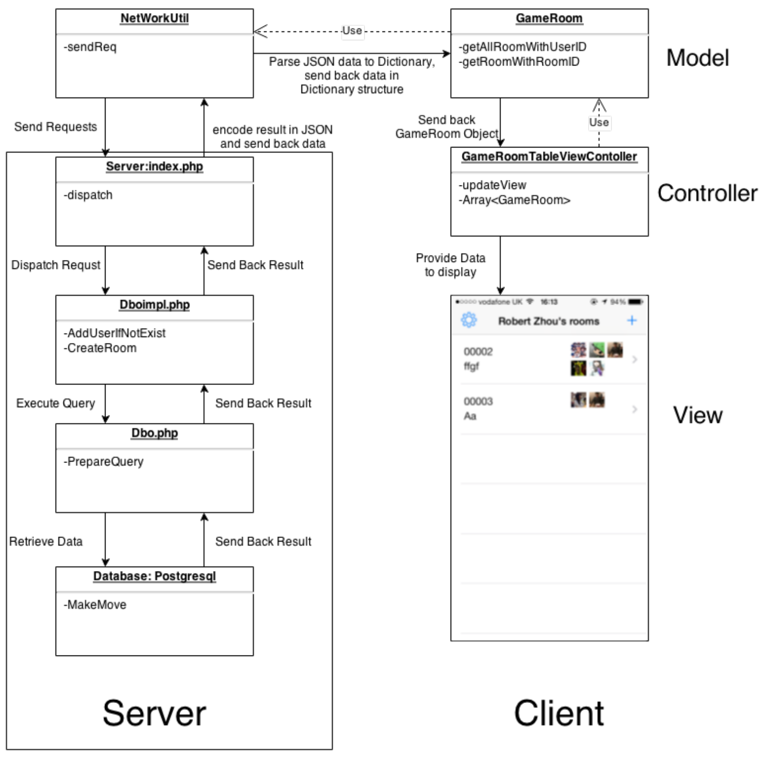
3.2 Design Practices

The front-end was implemented using the View-Model-Controller pattern, it was enforces by the iOS SDK and create a good separation of implementation, and allow the application to be easily modified and extend during the course of development. In the Front end, each window in the game are a view, i.e. The StationTableView, it is control by the StationTableViewController class, which will respond to user interaction, i.e. if the user tab on the "enter" button, it will display a new view to let the users see the available nearbt stations, and all the data/model was implemented by the GameRoom and GameStation class which is in charge of providing data and retrieve and send data to the server when necessary.

There is also a clear separation of front-end and back-end, all of the communication was implemented using the NetWorkUtil class, and data class such as the GameRoom will use NetWorkUtil to retrieve data, and the controller will only talk to the GameRoom class for game information, so each layer has a clear separated responsibility which is essential to minimise error during development.

In order to pass data, we have to establish a common data structure. We have chosen to use JavaScript Object Notation (JSON) which is a text-based open standard for exchanging and storing data. We can send a request to a PHP file and retrieve back data in JSON format. Likewise, Both Objective-c and PHP can receive and parse data in JSON format using built-in library functions provided, which makes communication very easy and efficient.

A good database structure goes a long way in supporting huge web project. We have written database schemas that are capable of supporting large number of players for our game and game information is divided and stored in different tables and not all store in a single table. Lastly, we also understand how we should distribute the load between client and the server. Calculations and data manipulations that can be done on the client's machine will be implemented using Javascript while those that can only be done on the server will use PHP.



3.3 Gameplay

The following section will include some screens shots to demonstrate how the game works, many of the interface are created during implementation stage, and they will be redesign and polished before the presentation.

The game is a native iphone app. The first page in the app is the facebook log in page. Once users register on the game by signing up, they enter the list of Game Room. The 'Game Room' page contains the list of stations. Players are to check in and check out whichever tube station they are nearby to.

3.4 Interface

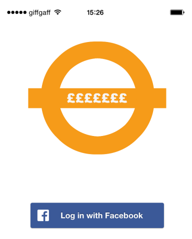


Figure 1: Log in Page

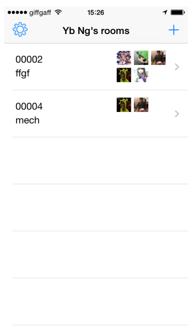


Figure 2: Game Room List View

When a user open the app, the first thing they see is the simplistic log in page with a facebook log in button. The log in process is taken care by facebook API. Upon logging in, the player will see the list of game rooms he has joined.

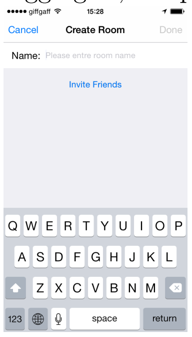


Figure 3: Create Room VIew

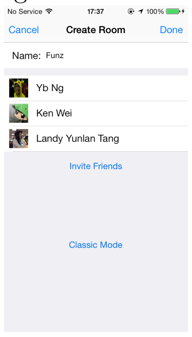


Figure 4

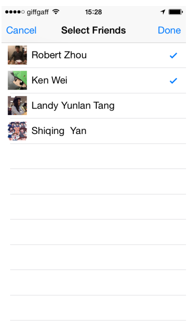


Figure 5: Select Friends View

If the user wants to create a new room, he can do so by clicking the "+" button on figure 2. The user can type in the room name and invite their facebook friends to join the room.

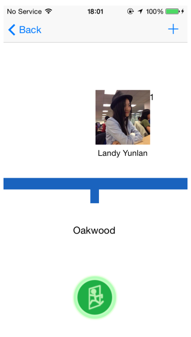


Figure 6: Station Map View

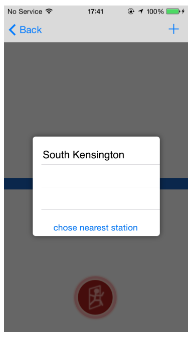


Figure 7: Station List View

When user enter the room, he will see a tube map view of his nearest station. If he wish to enter the station, he can tap on the green button. a selection of nearby stations will pop up. It is the same for exiting a station, other than having a red button instead.

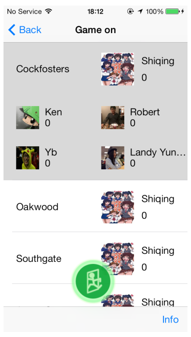
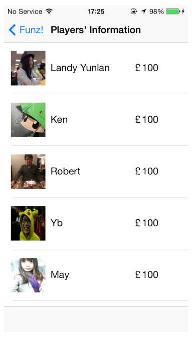


Figure 8: Room Table View



Landy Yunlan	￡100
Ken	￡100
Robert	￡100
Yb	￡100
May	￡100

Figure 9: Player Information View

The user can choose to see more information by selecting a table view (figure 8). Entering and exiting process will still be the same. In both view, the user can enter the player information view to see how other players in the room are performing.

3.5 Database and server

This game that we are developing rely heavy on the database. Therefore, database structure has to be designed with care. We store almost all player's information on the database.

4 Acknowledgements and Legal Issues

5 Conclusion

5.1 What we have done

We have successfully produced a simple multiplayer game using various web-programming languages and systems. Most of the targets we have specified were accomplished, although some features had to be simplified. We have managed to go through the entire process of a game development project from designing contents to implementation within the deadline. We have thought through the user interface design to make sure it is convenient for the players to use, and is appealing at the same time. We kept improving the design during the implementation and testing stage by playing the game and figuring out the aspects that could be irritating for the users or that could be improved to enhance user experience. As mentioned in the "Group structure" section, we have organised our group so that the work flow would be as efficient as possible. We communicated with each other continually to notify one another of the things we require from their part, and to discuss some considerations or issues that may arise during the development. We gained a valuable experience of working in a team, understanding weaknesses and strengths of each other, sharing opinions and accepting others' views (This assignment was different from other group projects that we have done so far in the sense that we were supposed to design what we need to do from the scratch. This allowed us to discuss more freely about our views and thoughts, and come to a conclusion with the gathered opinions). This has greatly opened up our imagination and we were able to think creatively without restriction. We are not only working on the project to get marks but we work on it because the game is fun to design and we have a passion to create the best game.

5.2 What we have not done

Had we given more time for the project, we would have considered more features and different style of gameplay. There are many things that we have come up with, but decided not to include in the final game due to the short implementation time given. These details would certainly improve the game and user experience, so if we were given enough time, we would have considered implementing those. Also, we could have tried to implement a web version or android version. Currently, the website for the game is developed and optimised on iphone, and we did not consider making it compatible with others . We could have gone through a more thorough testing on different platforms, as well as providing a way to adjust the size of the game interface for the machines with different screens.

5.3 What we have learnt

We have learnt how our knowledge in web-programming and server programming could be integrated together to form a working game that involves interaction between users. It was a great opportunity to learn more advanced use of the programming languages and to combine them with database manipulation. We have also learnt how big enterprise application design their database and adopt the same practice from them. This skills and knowledge is not easy to learn from classroom or lectures. Furthermore, this skills were picked up while we work on our project and with a passion to accomplish a great game, we were able to learn at a much quicker pace. In terms of managing the project, we have learnt how to time-frame the assignment appropriately so that we have enough time for each step of the development and deliver what we have initially set out to achieve by the deadline.

5.4 What we would do differently in the future

Using PHP and progreSQL seemed reasonable considering the short period of time assigned, and the features the chosen languages provide. We wanted to spend time on developing features rather than on learning too many new things, so we chose the ones that we were more familiar with instead of mysql. However, it would have been interesting to explore what other possibilities there are, and try out something completely new to see which one we would prefer if we were to start another similar development project.